Newland

PT 980-II

Application Migration Guide

(Migrating PT980-II application to PT30)

Revision History

Version	Description	Date
V1.0.0	Initial release.	October 24, 2015

Table of Contents

Chapter 1 About This Manual

Purpose

Most of PT980-II (hereinafter referred to as “the PT980-II”) applications can run on PT30 (hereinafter referred to as “the PT30”) smoothly and directly. However, due to differences in some APIs and hardware information between PT980-II and PT30 applications, some little PT980-II applications need to make some code modifications and rebuild in order to be compatible with PT30 according to this manual. This manual is the bible on how to migrate PT980-II applications to PT30, including things that shall be paid attention to, and how to set up your own programming & rebuilding environments.

Scope of Application

This manual is applicable to PT30 Standard (software version PT30_V3.00.021 or later required) and Lite (software version PT30_V4.00.004 or later required) Editions. If you have a PT30 Standard or Lite Edition with a software version older than what is required, you need to update it before migrating PT980-II applications to your PT30.

Besides, since PT30 Lite Edition is equipped with USI WiFi module while PT980-II with Summit WiFi module, migration of PT980-II application using Summit WiFi SDK to PT30 Lite Edition requires certain changes to be made to the corresponding API first.

Comparison of PT30 Standard and Lite Editions

PT30 Standard Edition	PT30 Lite Edition
Hardware version V3.x Software version V3.xx.xxx	Hardware version V4.x Software version V4.xx.xxx
Summit SSD30AG WiFi module supporting 802.11 a/b/g	USI WM-G-MR09 WiFi module supporting 802.11 b/g
BT module available	BT module not available
5V/1.5A power adapter	5V/2A power adapter

Chapter 2 PT30 Application Development Environment

Newland provides two SDKs for PT30 application development: **uTools For MT70_PT30** for C/C++ applications and **uToolsCE_NET** for .NET applications.

Installation and Use of uTools For MT70_PT30

uTools For MT70_PT30, which is add-on software based on Visual Studio 2005 or Visual Studio 2008, is a software development tool particularly designed for the creation of MT70/PT30 applications.

uTools For MT70_PT30 supports standard Windows CE APIs, integrates MFC library and provides visual GUIs, with which developers can develop C/C++ applications quickly and conveniently.

Installing uTools For MT70_PT30

◆ Download uTools For MT70_PT30 at **http://www.newlandaidc.com/h-col-116.html**.

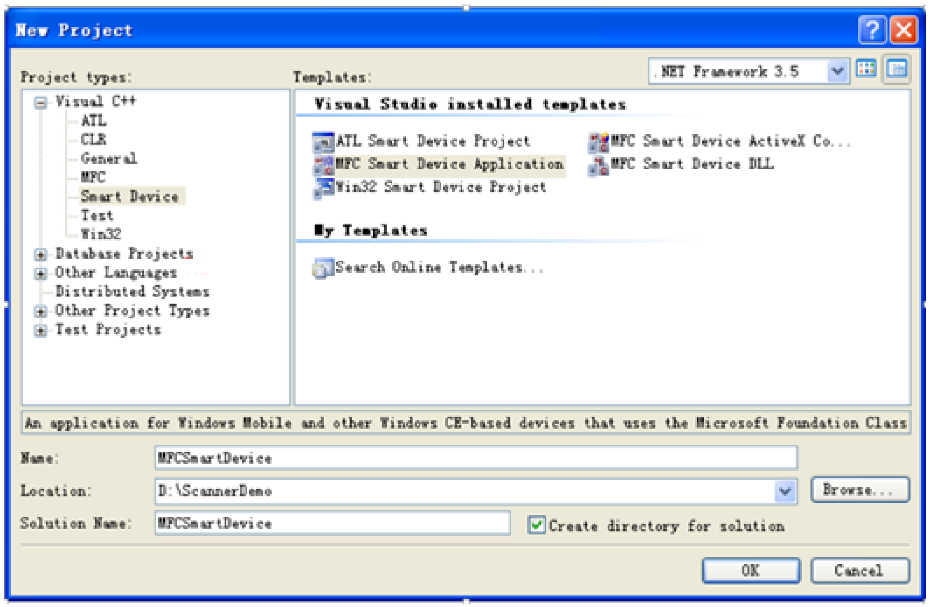
◆ Double-click on the file.

◆ Install the SDK in your desired location by following the on-screen instructions.

Setting up a Development Environment

Visual Studio 2008 is used in the example.

◆ Start Visual Studio 2008. Then create a new project as shown below.



An application for Windows Mobile and other Windows CE-based devices that uses the Microsoft Foundation Class

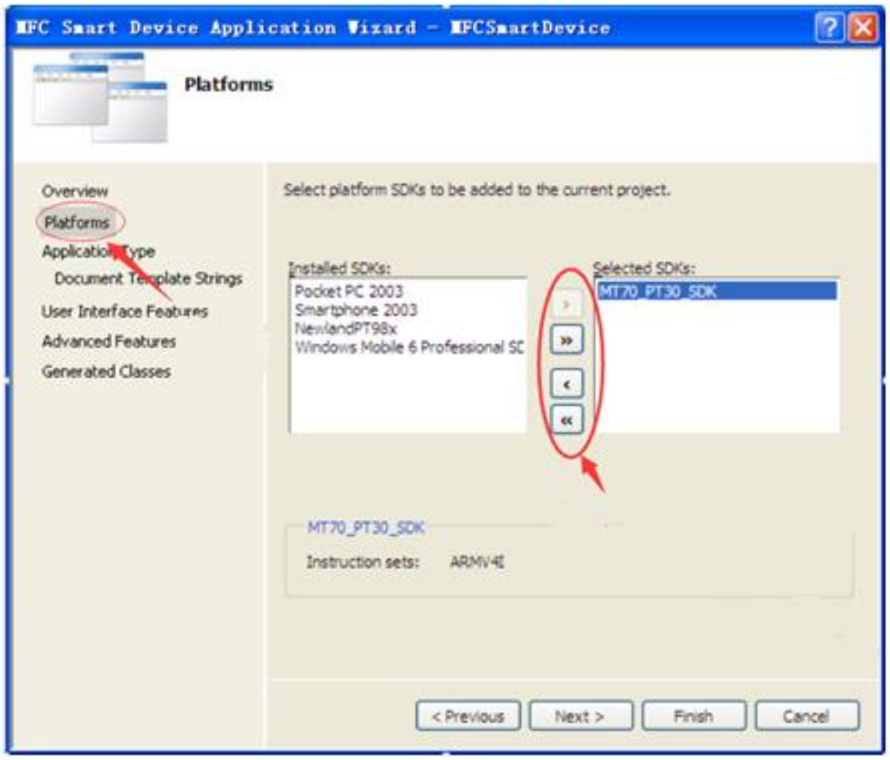
Name: MFCSmartDevice

Location: D:\ScannerDemo

Solution Name: MFCSmartDevice

Create directory for solution [✓]

◆ Select “MT70_PT30_SDK” on the “Platforms” page.



Select platform SDKs to be added to the current project.

Installed SDKs:
Pocket PC 2003
Smartphone 2003
NewlandPT98x
Windows Mobile 6 Professional SD

Selected SDKs:
MT70_PT30_SDK

MT70_PT30_SDK

Instruction sets: ARMV4I

Installation and Use of uToolsCE_NET

uToolsCE_NET is an additional .NET-related dynamic library set for Newland's Windows CE devices. It contains a dynamic library NLSCAN.MacCtrl.dll and demo programs. NLSCAN.MacCtrl.dll contains classes such as scan barcode, dial-up, get system info and system control.

Installing uToolsCE_NET

◆ Download uTools For MT70_PT30 at **http://www.newlandaidc.com/h-col-116.html**.

◆Unzip it and double-click on the file.

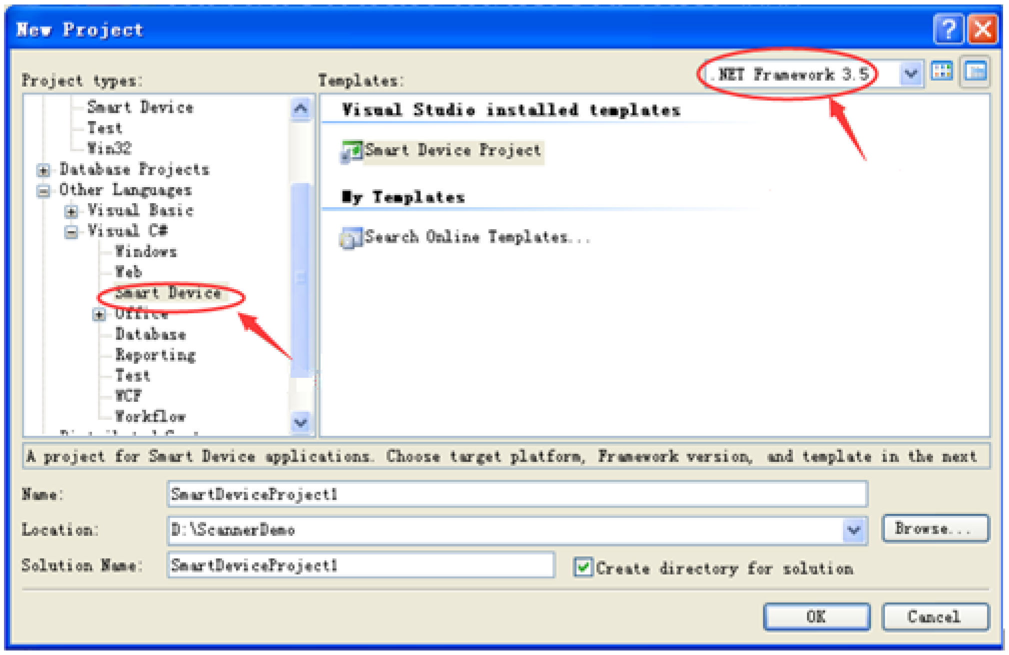
uToolsCE_NET-1.1...

◆ Install the SDK in your desired location by following the on-screen instructions.

Setting up a Development Environment

Visual Studio 2008 is used in the example.

◆ Start Visual Studio 2008. Then create a new project as shown below.



New Project

Project types:
- Smart Device
- Test
- Win32
- Database Projects
- Other Languages
  - Visual Basic
  - Visual C#
    - Windows
    - Web
    - Smart Device
    - Office
    - Database
    - Reporting
    - Test
    - WCF
    - Workflow

Templates: .NET Framework 3.5

Visual Studio installed templates
- Smart Device Project

My Templates
- Search Online Templates...

A project for Smart Device applications. Choose target platform, Framework version, and template in the next

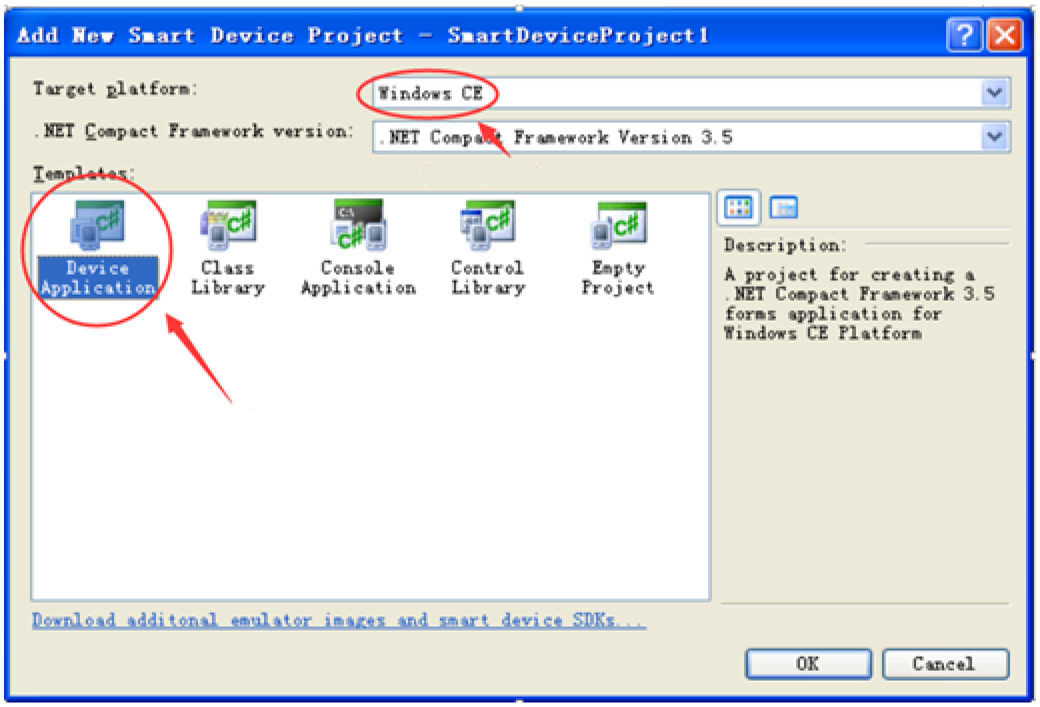
Name: SmartDeviceProject1

Location: D:\ScannerDemo

Solution Name: SmartDeviceProject1

☑ Create directory for solution

◆ Select the target platform and template as shown below.


Add New Smart Device Project - SmartDeviceProject1
Target platform:
.NET Compact Framework version:
Templates:
Device Application
Class Library
Console Application
Control Library
Empty Project
Download additonal emulator images and smart device SDKs...
OK
Cancel

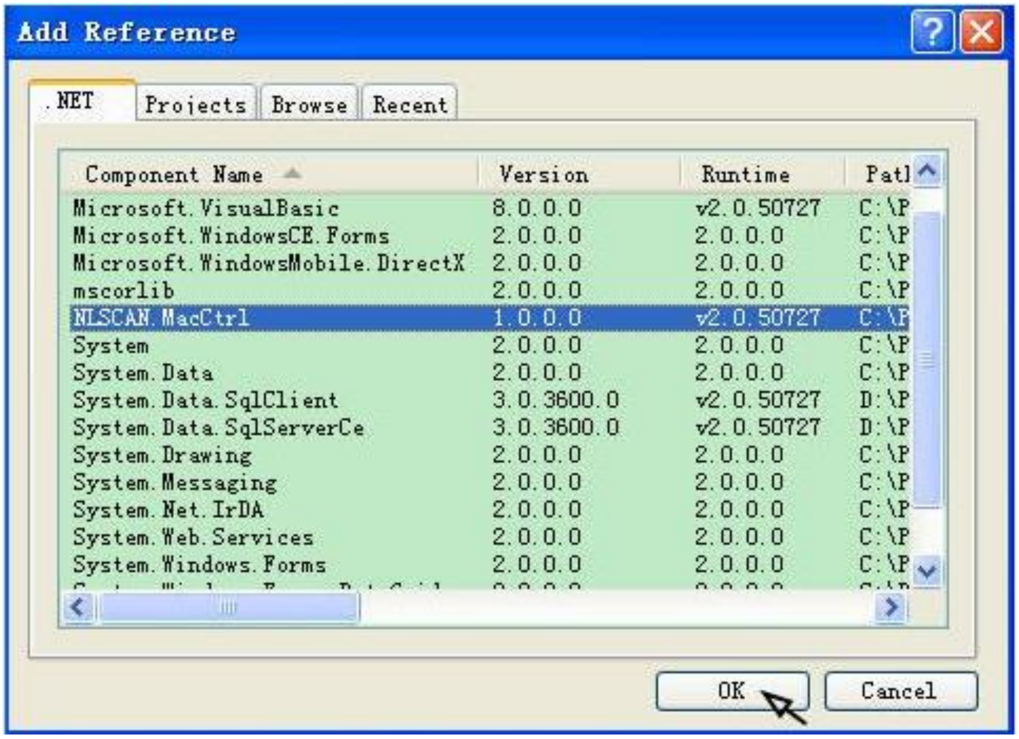
Target platform: Windows CE

.NET Compact Framework version: .NET Compact Framework Version 3.5

Description: A project for creating a .NET Compact Framework 3.5 forms application for Windows CE Platform

- Add reference.

Click **Menu** > **Project** > **Add Reference**. Then click the .NET tab, select “NLSCAN.MacCtrl” and click “OK”.



Component Name	Version	Runtime	Patl
Microsoft.VisualBasic	8.0.0.0	v2.0.50727	C:\P
Microsoft.WindowsCE.Forms	2.0.0.0	2.0.0.0	C:\P
Microsoft.WindowsMobile.DirectX	2.0.0.0	2.0.0.0	C:\P
mscorlib	2.0.0.0	2.0.0.0	C:\P
NLSCAN.MacCtrl	1.0.0.0	v2.0.50727	C:\P
System	2.0.0.0	2.0.0.0	C:\P
System.Data	2.0.0.0	2.0.0.0	C:\P
System.Data.SqlClient	3.0.3600.0	v2.0.50727	D:\P
System.Data.SqlServerCe	3.0.3600.0	v2.0.50727	D:\P
System.Drawing	2.0.0.0	2.0.0.0	C:\P
System.Messaging	2.0.0.0	2.0.0.0	C:\P
System.Net.IrDA	2.0.0.0	2.0.0.0	C:\P
System.Web.Services	2.0.0.0	2.0.0.0	C:\P
System.Windows.Forms	2.0.0.0	2.0.0.0	C:\P

Chapter 3 Code Migration Instructions

Code changes need to be made manually in the following situations when migrating PT980-II applications to the PT30.

Summit WiFi Code Migration

The PT980-II uses the API provided by Summit SDK to manage WiFi functionality whereas PT30 Lite Edition uses WinCE's standard WiFi API, so the API needs to be changed accordingly when migrating PT980-II's Summit WiFi code to PT30 Lite Edition.

WZC (Wireless Zero Configuration) functions can be used to connect PT30 Lite Edition to a WiFi network.

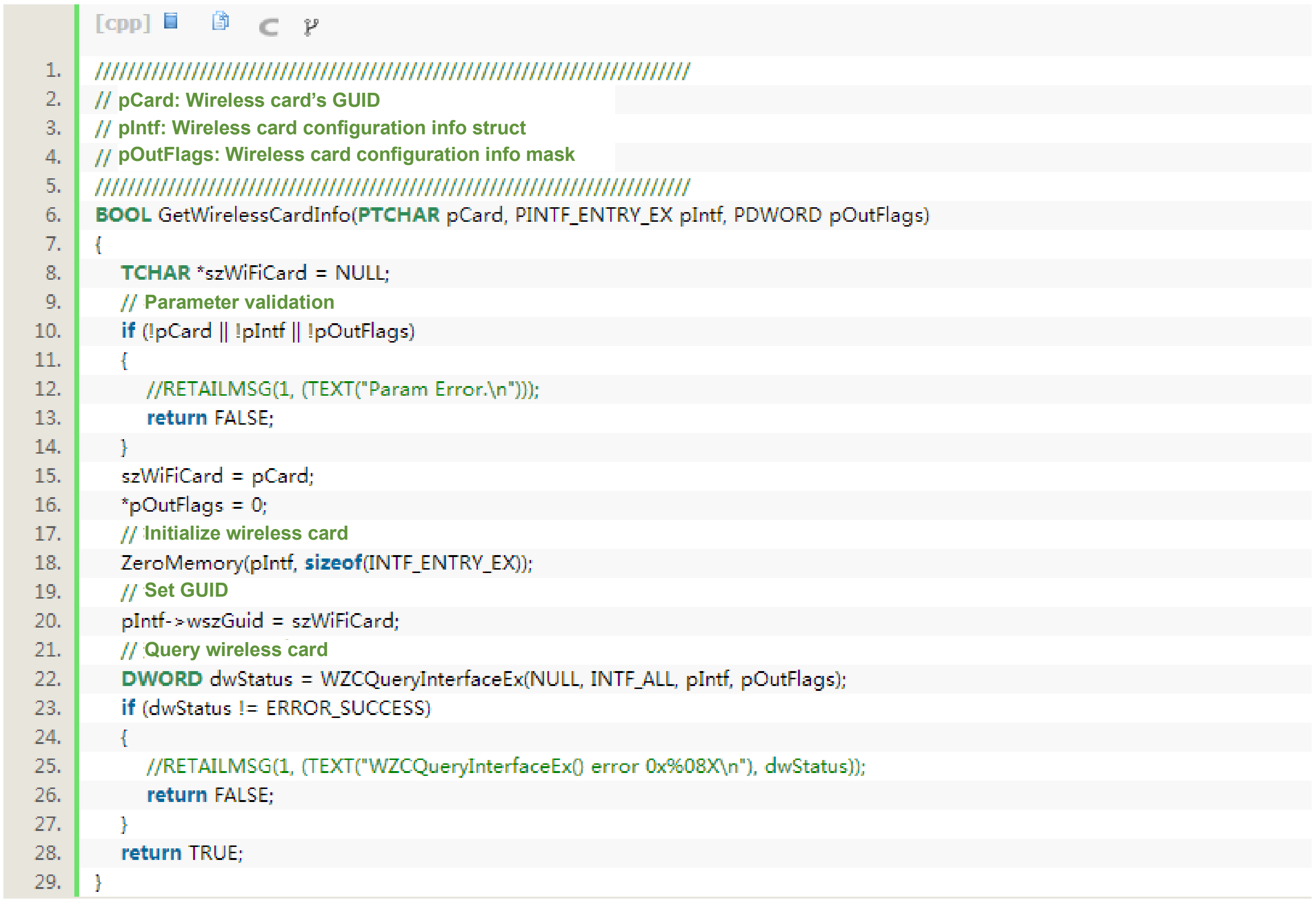
Steps required to connect PT30 Lite Edition to a WiFi network: First, get PT30's wireless card info; second, get current wireless network info; third, pass the SSID obtained and password entered to the preferred wireless network list, then PT30 Lite Edition will automatically connect to that wireless network.

Frequently used WZC functions include:

Function	**Description**
WZCEnumInterfaces	Determine the identity of wireless card in the terminal
WZCPassword2Key	Translate a user password into a 256-bit network key.
WZCQueryInterface	Provide detailed information for the specified wireless card
WZCDeleteIntfObj	Release memory associated from a corresponding call to WZCQueryInterface

For the information of other API functions, see
https://msdn.microsoft.com/en-us/library/ee486655(v=winembedded.60).aspx

The following sample code illustrates how to get wireless network information from wireless card.



```cpp
///////////////////////////////////////////////////////////////////////////
// pCard: Wireless card's GUID
// pIntf: Wireless card configuration info struct
// pOutFlags: Wireless card configuration info mask
///////////////////////////////////////////////////////////////////////////
BOOL GetWirelessCardInfo(PTCHAR pCard, PINTF_ENTRY_EX pIntf, PDWORD pOutFlags)
{
    TCHAR *szWiFiCard = NULL;
    // Parameter validation
    if (!pCard || !pIntf || !pOutFlags)
    {
        //RETAILMSG(1, (TEXT("Param Error.\n")));
        return FALSE;
    }
    szWiFiCard = pCard;
    *pOutFlags = 0;
    // Initialize wireless card
    ZeroMemory(pIntf, sizeof(INTF_ENTRY_EX));
    // Set GUID
    pIntf->wszGuid = szWiFiCard;
    // Query wireless card
    DWORD dwStatus = WZCQueryInterfaceEx(NULL, INTF_ALL, pIntf, pOutFlags);
    if (dwStatus != ERROR_SUCCESS)
    {
        //RETAILMSG(1, (TEXT("WZCQueryInterfaceEx() error 0x%08X\n"), dwStatus));
        return FALSE;
    }
    return TRUE;
}
```

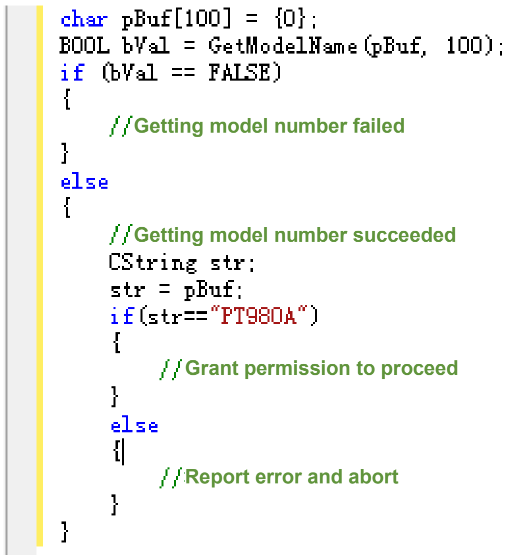
Hardware-related Code Migration

The hardware information (such as model number, serial number and MAC address) of PT980-II is different from that of PT30. For PT980-II applications that take certain action based on judgment concerning hardware information obtained (e.g. grant/ deny permission to proceed if model number obtained is/ is not PT980-II), the code needs to be changed accordingly when migrating those applications to the PT30.

When developing application using C++, ***GetModelName*** can be used to obtain model number.

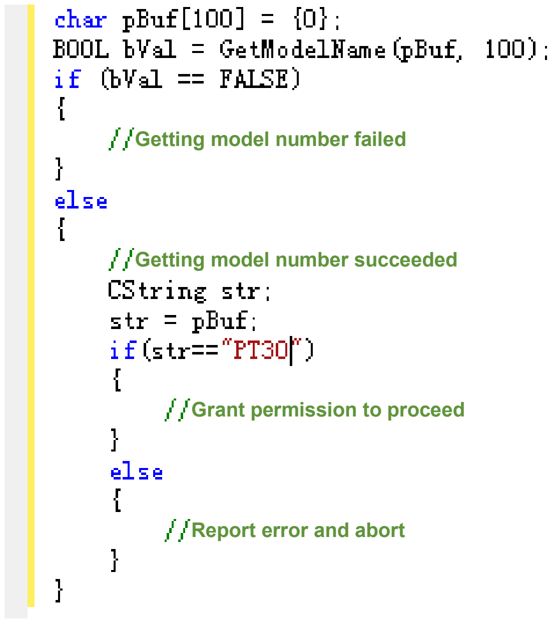
Example:

Original code of the PT980-II:



```
char pBuf[100] = {0};
BOOL bVal = GetModelName(pBuf, 100);
if (bVal == FALSE)
{
    //Getting model number failed
}
else
{
    //Getting model number succeeded
    CString str;
    str = pBuf;
    if(str=="PT980A")
    {
        //Grant permission to proceed
    }
    else
    {
        //Report error and abort
    }
}
```

Modified code for the PT30:



```
char pBuf[100] = {0};
BOOL bVal = GetModelName(pBuf, 100);
if (bVal == FALSE)
{
    //Getting model number failed
}
else
{
    //Getting model number succeeded
    CString str;
    str = pBuf;
    if(str=="PT30")
    {
        //Grant permission to proceed
    }
    else
    {
        //Report error and abort
    }
}
```

Keyboard-related Code Migration

The keys that exist on PT980-II keyboard but cannot be found on PT30 keyboard include *****, **#** and **Fn** keys. For PT980-II applications that involve any of those three keys, such keys need to be redefined according to PT30 keyboard when migrating those applications to the PT30.

There are two methods for redefining keys.

Method One: Modify the value of key in the registry

Step 1: Open the registry of the PT980-II and locate the *HKLM\PDA\KeyMap\VKey* node. If *HKLM\PDA\KeyMap\VKey* does not exist, create it.

Open the registry on PC: Connect the PT980-II to the PT30 with a USB cable, then click **Start** -> **Programs** -> **Microsoft Visual Studio 2005** -> **Visual Studio Remote Tools** -> **Remote Registry Editor** on PC.

Open the registry on PT980-II: On the terminal, click **Start** -> **Run** and then enter “regedit” and press the Enter key.

Step 2: Under the *HKLM\PDA\KeyMap\VKey* node add or modify the registry item(s) you want: data type is DWORD; valueName (See the following table); data is the virtual key value of physical key.

Keyboard	valueName
Key on the left side	“LeftSide”
Key on the right side	“RightSide”
OK key	“OK”
Cancel key	“Cancel”
Left key	“Left”
Right key	“Right”
Up key	“Up”
Down key	“Down”
SCAN key	“Scan”
Backspace key	“BackSpace”
Space key	“Space”
Fn key	“Fn”
* key	“*”
# key	“#”
1 key	“1”
2 key	“2”
3 key	“3”
4 key	“4”
5 key	“5”
6 key	“6”
7 key	“7”
8 key	“8”
9 key	“9”
0 key	“0”

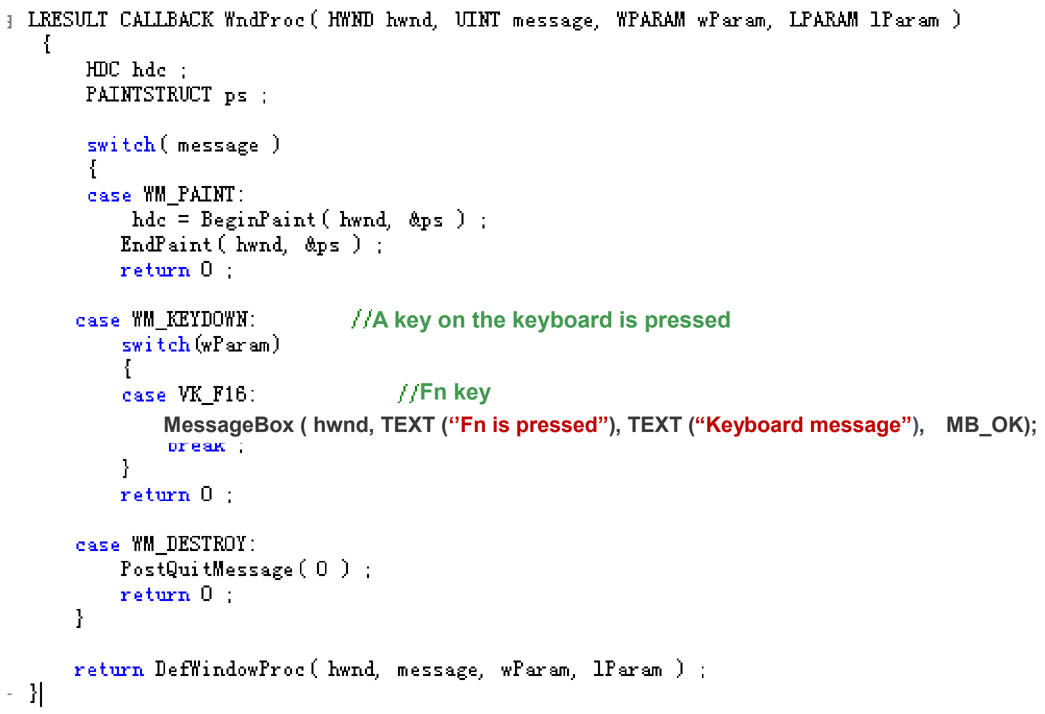
Example:

To define * key on the keyboard as **.** key, add or modify the registry item “*” (value is 190, data type is DWORD). The virtual key value of **.** key is 190. For the virtual key value of each physical key, refer to MSDN.

Method Two: Modify the code

For example, a PT980-II application uses **Fn** key to trigger an action, but **Fn** key is not available on PT30 keyboard, so **Fn** key is replaced by **1** key in the PT980-II application when migrating the application to PT30.

Original code of the PT980-II:



```
LRESULT CALLBACK WndProc( HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam )
 {
     HDC hdc ;
     PAINTSTRUCT ps ;

     switch( message )
     {
     case WM_PAINT:
         hdc = BeginPaint( hwnd, &ps ) ;
        EndPaint( hwnd, &ps ) ;
        return 0 ;

    case WM_KEYDOWN:          //A key on the keyboard is pressed
        switch(wParam)
        {
        case VK_F16:             //Fn key
            MessageBox ( hwnd, TEXT (“Fn is pressed”), TEXT (“Keyboard message”),   MB_OK);
            break ;
        }
        return 0 ;

    case WM_DESTROY:
        PostQuitMessage( 0 ) ;
        return 0 ;
    }

    return DefWindowProc( hwnd, message, wParam, lParam ) ;
 }
```

Modified code for the PT30:

```
LRESULT CALLBACK WndProc( HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam )
 {
      HDC hdc ;
      PAINTSTRUCT ps ;

      switch( message )
      {
      case WM_PAINT:
          hdc = BeginPaint( hwnd, &ps ) ;
         EndPaint( hwnd, &ps ) ;
         return 0 ;

     case WM_KEYDOWN:          //A key on the keyboard is pressed
         switch(wParam)
         {
         case '1':               //1 key
             MessageBox ( hwnd, TEXT ("1 is pressed"), TEXT ("Keyboard message"),   MB_OK);
             break ;
         }
         return 0 ;

     case WM_DESTROY:
         PostQuitMessage( 0 ) ;
         return 0 ;
     }

     return DefWindowProc( hwnd, message, wParam, lParam ) ;
 }
```

Summary: Method one is simple-to-use, but registry changes will affect all applications involving changed keys. If you only want the changed keys to be used in a specific application, use method two.

Others

(1) The PT980-II generates sounds with a buzzer whereas the PT30 generates sounds with a speaker. The buzzer control function ***buz_ctrl*** used by the PT980-II can execute on PT30 Lite Edition, but the tone and frequency of sound generated will be different. So adjustments are required accordingly.

(2) There are six levels of keyboard backlighting for the PT980-II whereas there are only two levels (On and Off) for PT30 Lite Edition. The keyboard backlight control function ***kbd_bkl_set*** used by the PT980-II can execute on PT30 Lite Edition, but there will be two levels available.

Newland EMEA
+31 (0) 345 87 00 33
info@newland-id.com

Newland D-A-CH
+49 (0) 6182 82916-16
info@newland-id.de

Newland UK
+44 (0) 1442 212 020
sales@newland-id.co.uk

Newland Nordic
+46 (0) 708 847 767
nordic@newland-id.com

Newland Ibérica
+34 (0) 93 303 74 66
info@newland-id.es

Newland France
+31 (0) 345 87 00 33
france@newland-id.com

Newland Middle East
+31 (0) 345 87 00 33
middleeast@newland-id.com

Newland South Africa
+27 (0) 11 553 8010
info@newland-id.co.za

Newland Italy
+39 (0) 342 056 2227
italy@newland-id.com

Newland Russia
+31 (0) 345 87 00 33
russia@newland-id.com

Newland Turkey & Iran
+90 (0) 544 538 40 49
turkey@newland-id.com
iran@newland-id.com